T-79.514 Special Course on Cryptology / Privacy-preserving K-means clustering

Tatiana Issaeva

November 1, 2003

1 Overview

The paper under review by Vaidya and Clifton [1] deals with the K-means clustering data. K-means clustering is a data mining technique, which divides data into K groups (clusters) according to data's positioning to each other.

Authors provide algorithms for running the K-means in multiparty mode in a privacy-preserving fashion under the assumption, that different parties contain different data for the same entities.

2 Introduction

Clustering techniques are widely used in many mathematical and technical disciplines for finding groups of items with similar parameters. K-means clustering is a technique that divides data into K clusters placed in a multidimensional space around K means. The dimension of the space is the number of the attributes that are being used for clustering. A cluster contains items whose attribute values are similar.

There is nothing complicated to clustering of data. The clustering technique is the same if one needs to cluster data located on multiple parties, only the communication aspect makes the task more demanding. The communication of data between the parties is the main problem authors of this paper tried to solve. The communicated data means that other parties learn about specific details of each others data. If the communicated data is in any way private, like bank records, the data communication becomes a privacy violating exercise.

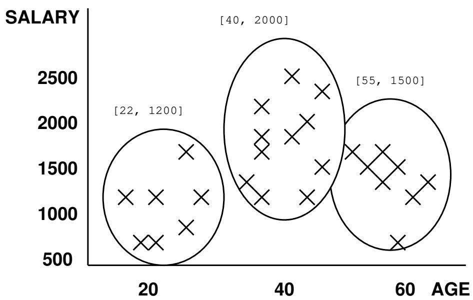
Authors tailored their paper on the data mining between several parties. Each party has different set of attributes for the same set of entities, e.g. for each citizen the banks can have account and transaction information, the police can have criminal records and the health service can hold health records. Such data is called vertically partitioned. Direct exchange of the data between these entities is impossible as it violates one's right to privacy and exposed party's data to other parties..

The proposed privacy-preserving clustering modifies the clustering algorithm and the data communication heavily to withold each party's data from other sites but at the same time allow for multisite clustering.

3 System details

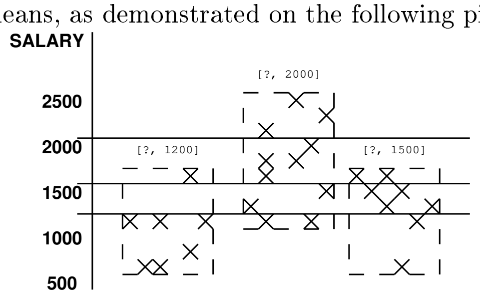
3.1 K-means clustering

K-means clustering groups items into L groups according to L attributes. The number of attributes L creates a L-dimensional space in which each item is located based on the values of its attributes. E.g. when clustering by person's age and salary, we create two-dimensional space with axes of age and salary. We can then cluster the data into K clusters, so that each person will at the end belong into one of the K clusters which are centered around K means (cluster centers - they define a cluster, not its shape). The way clusters will form depends on the initial cluster coordinates and on the data we have (e.g. no young and high salaried people - no such cluster).



3.2 Privacy-preserving K-means clustering

The task here is to cluster data from R parties, each having different attributes for the same set of entities. The parties wish to cluster their joint data using the K-means algorithms without disclosing its own data to other parties. The result of the clustering would be positions of the clusters with each party receiving only the relevant part of the cluster means according to the attributes the party contributed to the computation. E.g. the party contributing the salary attributes will receive only the salary coordinates of the means, as demonstrated on the following picture:



Each party contributes only its own data and receives only its own portion of the means' coordinates and assignment of entities to clusters so the privacy of others' data is preserved. This is the idea, now let's have a look how it can be done.

To provide privacy, the parties do not communicate the actual data at all. Each party computes the distance between its data and its part of clusters' means locally and communicates only the results. The transmission of the distances vector is never in plaintext but is always encrypted by homomorphic encryption scheme. The elements of the distances vector are further permuted (scrambled) to hide actual values placement.

The conditions just described above would not give any means for computations at other parties as the data would be encrypted and scrambled, so the algorithm depends on three parties, that take specific roles in the K-means computation. They can be either the parties holding the data or completely external parties introduced just for the purpose of secure K-means clustering computation. In either case they must not cooperate, otherwise the actual data can be exposed by combining special sites' knowledge of the scrambling and encryption parameters.

3.3 Party P_1 as the initiator

The paper introduces the special party P_1, which is the initiator of the K-means computation and which is also decrypting and distributing the real results at the end of the computation.

Party P_1 firstly generated separate vector V_i for each party. Each vector V_i has dimension K (number of desired clusters) and the sum of all the V_i vectors is a null vector $\sum_1^K V_i = 0$. Therefore the elements in each dimension of the vectors V_i have pseudo random distribution. These vectors are to be used as random shift operators of party's own data - party sums its vector of distances X_i with the random vector X_i and the resulting vector of data makes no sense to anybody not knowing the original vector V_i.

Party P_1 also generates permutation π that is the same for each party, used to further mangle the data by permuting each party's vector :$\pi(X_i + V_i)$. This way the data X_i is shifted $(X_i + V_i)$ and also scrambled $\pi(X_i + V_i)$.

Interesting part that comes to play now is the homomorphic encryption function - H. Examples of such a function can be found in e.g. [2], [3] or [4]. This function is a public key encryption function. What makes it homomorphic is the fact, that there exists an efficient way to compute $H(x + y)$ from $H(x)$ and $H(y)$ without revealing the neither x nor y. Without more details, let's just take any such homomorphic function H and use it in a permutation part of the K-means algorithm.

Use of the homomorphic function allows for not actually revealing the vector V_i to its party and not revealing the permutation π to any party - they are both stay safe with its creator - P_1. Using the public-key encryption of the H function, the P_1 and other parties exchange only encrypted vectors V_i and X_i. Each party encrypts its own X_i vector and sends it to P_1 and receives $\pi H(X_i^{'} + V_i^{'})$ - permuted sum of encrypted vectors, from which it can decrypt the actual permuted sum $\pi(X_i + V_i)$. After such a setup the following transmissions will be secure.

3.4 Parties P_2 and P_r - computing the distances

Special parties P_2 and P_r (there are r-parties involved in the K-means computation) get involved when the we actually run the K-means algorithm. All parties except P_2 and P_r now send their vectors to P_r. The data being send by each party is its own permuted vector of the sum : $\pi(X_i+V_i)$. Note that each party does not know its vector V_i or common permutation π, they just know the result $\pi(X_i+V_i)$ from the initial interaction with P_1.

Now the parties P_2 and P_r will engage in a secure addition and comparison operation to find out the closest cluster. As P_r now has the vectors from all parties except P_2, and since the sum of all the V_i vectors gives a null vector, party P_r can in cooperation with P_2 find the closest cluster. This is done by a secure circuit evaluation procedure, generally introduced in [5] and described in [6]. The secure circuit computation in our case is the secure_add_and_compare operation denoted as $a_2 + a_r < b_2 + b_r$. The aim of the secure circuit computation is to disclose nothing but the comparison result. Parties P_r and P_2 now have to make $K - 1$ such secure multiparty comparisons to determine the closest cluster.

3.5 Party P_1 as the result distributor

After all the comparisons, parties P_r and P_2 know the closest cluster in the permutation π. Because the actual permutation π is not known to them, they have to send their result (i - the index of the closest cluster) back to P_1. Party P_1 removes the permutation from the i-th cluster and broadcasts the results as the proper cluster for the computed entity.

4 Handling collusion

It should be now apparent that special parties P_1, P_2 and especially P_r learn quite much during the computation process. Authors prove in chapter three of their paper [1] in a formal way that the information learned by the special parties is meaningless in isolation.

However if the parties P_2 and P_r would collude they could easily derive the distances between the points and the means for each party. Therefore the parties P_2 and P_r must be selected very carefully so that it is in their own interest to not collude.

5 Communication costs

The total communication cost is dependent on the number of iterations required for the K-means algorithm to converge and this is unpredictable. On the other hand the communication costs of one round of computation is predictable and incures in the initial permutation algorithm, in the secure_add_and_compare algorithm and the closes cluster computation.

Chapter five of the paper [1] gives details of the communication cost computations. The cost of one iteration of the K-means algorithm requires $O(r+k)$ rounds of communication, each with cost $O(nrk)$ bits, where r is the number of parties, k is the number of desired clusters and n is the number of entities to be clustered.

6 Conclusion

The method for privacy-preserving multiparty K-means clustering computation is given. The paper explains well the basic ideas behind the K-means clustering and defines the privacy problem for the multiparty computation. Authors propose a privacy-preserving scheme under which the computation of the K-means algorithm is centralized at three special non-colluding parties, data is transmitted as encrypted and scrambled and results of the clustering do not reveal other parties' data. Security proof for the multi-party computation, cluster computation and overall K-means is presented.

The security of this solution is based on the assumption that the special parties P_1, P_2 and P_r will not cooperate to gain other data of parties. In fact, the cooperation of the parties P_2 and P_r is enough to reveal the specific data of the other parties. The security aspects add additional load of communication and computation and this is discussed in the paper and optimizations are proposed.

Unfortunately the description of the later stages of the algorithm is not very specific and more information would be needed for better understanding and assessment. For example the actions of P_1 at the distribution of the results is not described well at all. The requirement for three special parties P_1, P_2 and P_r - either selected from the data holding parties or outsiders - limits usability of this solution to larger scale multiparty computations.

References

[1] Jaideep Vaidya, Chris Clifton. Privacy-Preserving K-Means Clustering over Vertically Partitioned Data. In Proceeedings of the Ninth ACK SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM 2003.

[2] J. Benaloh. Dense probabilistic encryption. In Proceedings of the Workshop on Selected Areas of Cryptography, pages 120-128, Kingston, Ontario, May 1994.

[3] D. Nacache and J. Stern. A new public key cryptosystem based on higher residues. In Proceedings of the 5th ACM conference on Computer and Communications Security, pages 59-66, San Francisco, California, United States, 1998, ACM Press.

[4] T. Okamoto and S. Uchiyama. A new public key cryptosystem as secure as factoring. In Advances in Cryptology - Eurorypt '98, LNCS 1403, pages 308-318. Springer-Verlag, 1998.

[5] A. C. Yao. How to generate and exchange secrets. In Proceedings of the 27th IEEE Symposium on Foundations of Computer Science, pages 162-167. IEEE, 1986.

[6] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. In 19th ACM Symposium on the Theory of Computing, pages 218-229, 1987.